

“Day 3 Integration Report”

Api Integration and Data migration

Table of Contents

1. Setting Up Environment Variables
2. Obtaining Sanity Project ID and API Token
3. Creating the Sanity Schema
4. Setting Up the Data Import Script
5. Running the Import Script

1. Setting Up Environment Variables

First, let's set up our environment variables. Create a ``.env.local`` file in your project root if it doesn't already exist. Add the following variables:

```
NEXT_PUBLIC_SANITY_PROJECT_ID=your_project_id
NEXT_PUBLIC_SANITY_DATASET=production
SANITY_API_TOKEN=your_sanity_token
```

Remember, variables prefixed with `NEXT_PUBLIC_` will be exposed to the browser, so be cautious about what you prefix.

2. Obtaining Sanity Project ID and API Token

Project ID

To find your Sanity project ID:

1. Log in to your Sanity account at

<https://www.sanity.io/manage>

2. Select your project

3. In the project dashboard, you'll see the project ID listed



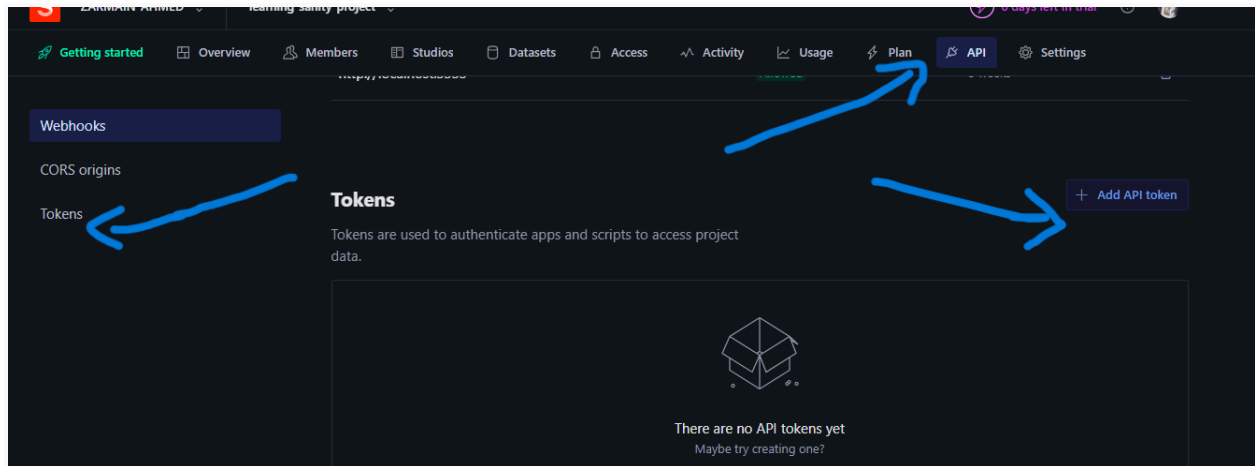
Use this ID for the `NEXT_PUBLIC_SANITY_PROJECT_ID` in your `.env.local` file.

API Token

To generate a Sanity API token:

1. Go to <https://www.sanity.io/manage> and select your project
2. Navigate to the "API" tab
3. Under "Tokens," click "Add API token"
4. Give your token a name and select the appropriate permissions (usually "Editor" for full read/write access)

5. Copy the generated token



Tokens

Tokens are used to authenticate apps and scripts to access project data.

Name

Examples: "Employee import", "Website preview" or "PDF generator".

revalidate

Permissions

Choose the access privileges for the token.

Contributor

☐ Read and write access to draft content within all datasets, with no access to project settings. (Tokens: read+write drafts)

Deploy Studio (Token only)

☐ Access to deploy Sanity Studio and GraphQL APIs to our hosted service.

Developer

☐ Read and write access to all datasets, with access to project settings for developers. (Tokens: read+write)

Editor

☒ Read and write access to all datasets, with limited access to project settings. (Tokens: read+write)

Viewer

☐ Read access to all datasets, with limited access to project settings. (Tokens: read-only)

Save Cancel

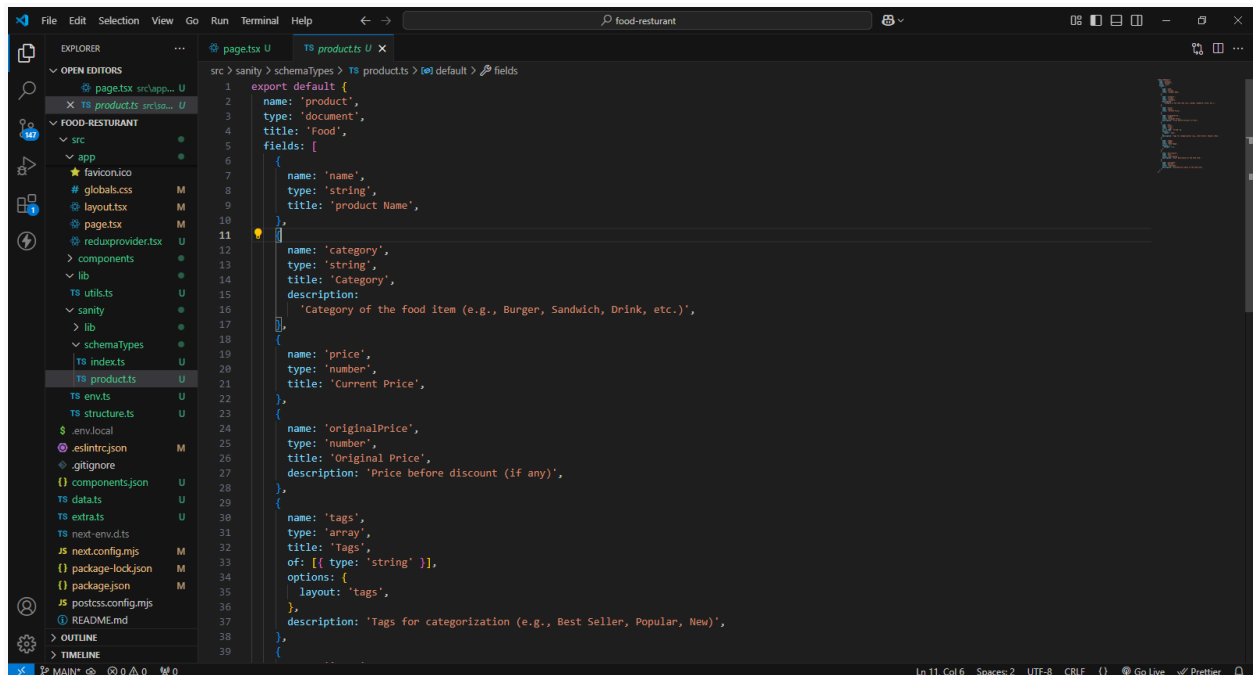
The screenshot shows the GitHub 'Tokens' page. On the left sidebar, 'Webhooks' is selected. The main content area is titled 'Tokens' and includes a '+ Add API token' button. Below this, a table lists existing tokens. One token named 'revalidate' is shown with 'Editor' permissions and a creation time of 'just now'. A red rectangular box highlights the token value field, which contains a long alphanumeric string. This string is completely obscured by a large, thick blue scribble. To the right of the token value, there is a small square icon with a document symbol, which is used to copy the token. A red arrow points from the right side of the image towards this copy icon.

NAME	PERMISSIONS	CREATED
revalidate	Editor	just now

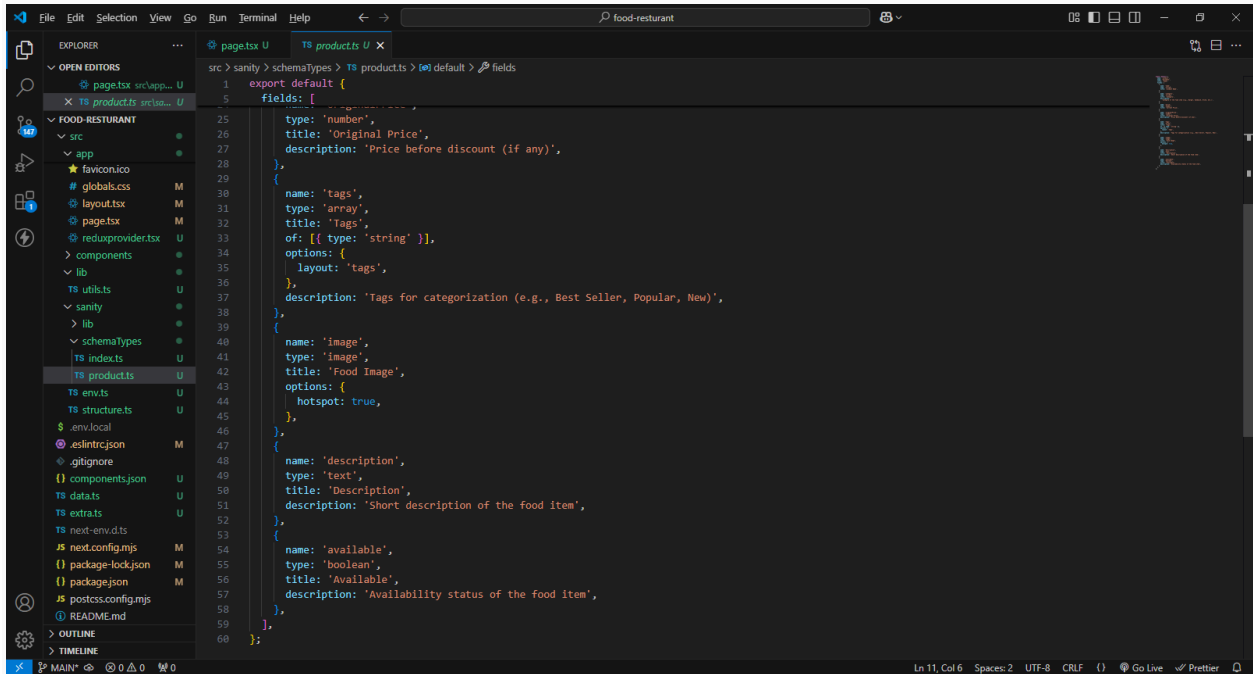
Use this token for the `SANITY_API_TOKEN` in your `.env.local` file.**3.**

Creating the Sanity Schema

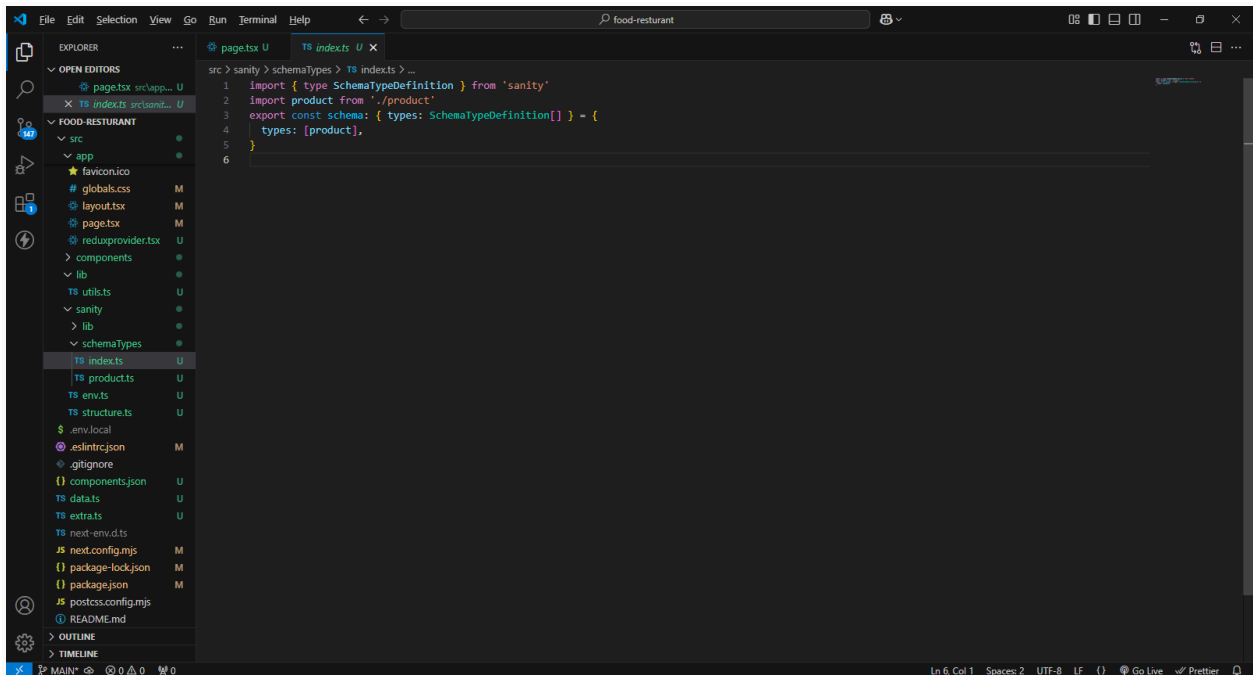
Now, let's create a schema for our products. In your Sanity schema folder (usually `sanity/schemaTypes`), create a new file called `product.ts`:



```
1 export default {
2   name: 'product',
3   type: 'document',
4   title: 'Food',
5   fields: [
6     {
7       name: 'name',
8       type: 'string',
9       title: 'product Name',
10    },
11  ],
12  {
13    name: 'category',
14    type: 'string',
15    title: 'Category',
16    description:
17      'Category of the food item (e.g., Burger, Sandwich, Drink, etc.)',
18  },
19  {
20    name: 'price',
21    type: 'number',
22    title: 'Current Price',
23  },
24  {
25    name: 'originalPrice',
26    type: 'number',
27    title: 'Original Price',
28    description: 'Price before discount (if any)',
29  },
30  {
31    name: 'tags',
32    type: 'array',
33    title: 'Tags',
34    of: [{ type: 'string' }],
35    options: {
36      layout: 'tags',
37    },
38    description: 'Tags for categorization (e.g., Best Seller, Popular, New)',
39  },
40  ],
41 }
```

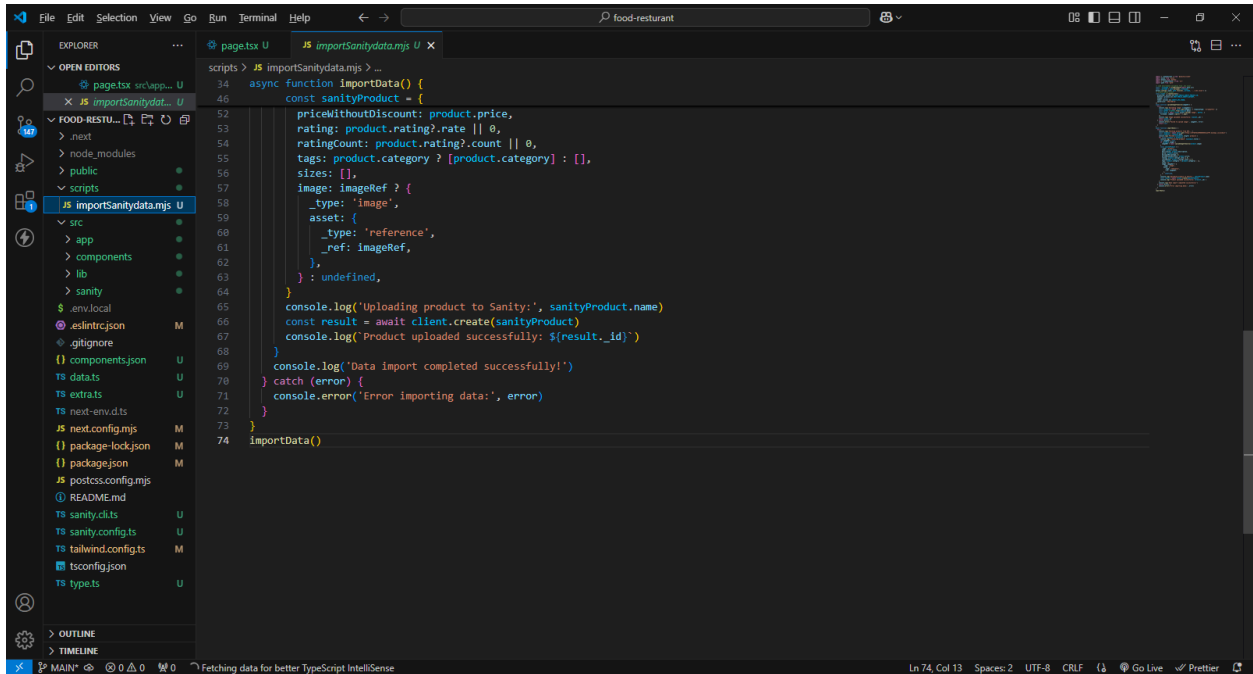


Then, update your `sanity/schemas/index.ts` file to include the new product schema:



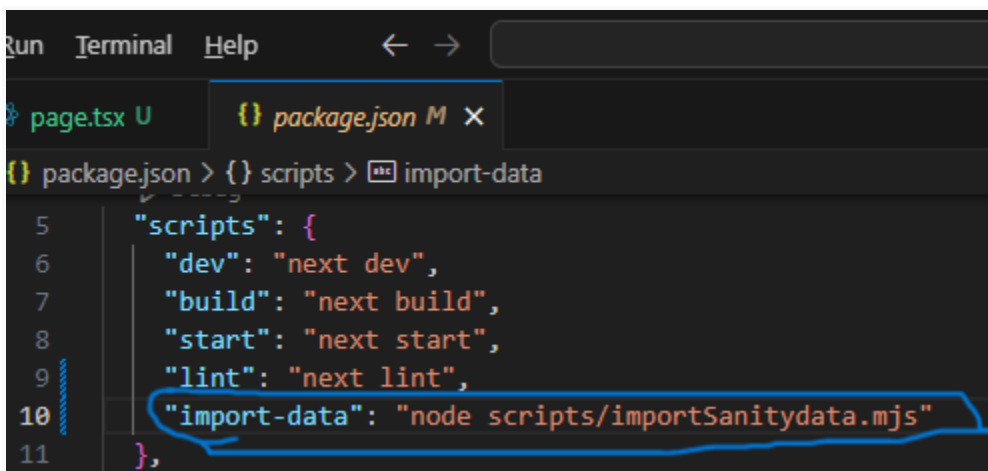
```
1 import { createClient } from '@sanity/client'
2 import axios from 'axios'
3 import dotenv from 'dotenv'
4 import { fileURLToPath } from 'url'
5 import path from 'path'
6
7 // Load environment variables from .env.local
8 const filename = fileURLToPath(import.meta.url)
9 const dirname = path.dirname(filename)
10 dotenv.config({ path: path.resolve(dirname, '../.env.local') })
11
12 // Create Sanity client
13 const client = createClient({
14   projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
15   dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
16   useCdn: false,
17   token: process.env.SANITY_API_TOKEN,
18   apiVersion: '2021-08-31'
19 })
20
21 async function uploadImageToSanity(imageUrl) {
22   try {
23     console.log('Uploading image: ${imageUrl}')
24     const response = await axios.get(imageUrl, { responseType: 'arraybuffer' })
25     const buffer = Buffer.from(response.data)
26     const asset = await client.assets.upload('image', buffer, {
27       filename: imageUrl.split('/').pop()
28     })
29     console.log('Image uploaded successfully: ${asset.id}')
30     return asset.id
31   } catch (error) {
32     console.error('Failed to upload image:', imageUrl, error)
33     return null
34   }
35 }
36
37 async function importData() {
38   try {
39     console.log('Fetching products from API...')
```

```
19 async function uploadImageToSanity(imageUrl) {
20   try {
21     console.log('Uploading image: ${imageUrl}')
22     const response = await axios.get(imageUrl, { responseType: 'arraybuffer' })
23     const buffer = Buffer.from(response.data)
24     const asset = await client.assets.upload('image', buffer, {
25       filename: imageUrl.split('/').pop()
26     })
27     console.log('Image uploaded successfully: ${asset.id}')
28     return asset.id
29   } catch (error) {
30     console.error('Failed to upload image:', imageUrl, error)
31     return null
32   }
33 }
34
35 async function importData() {
36   try {
37     console.log('Fetching products from API...')
38     const response = await axios.get('https://677d57d14496848554ca2f9f.mockapi.io/product')
39     const products = response.data
40     console.log('Fetched ${products.length} products')
41     for (const product of products) {
42       console.log('Processing product: ${product.title}')
43       let imageRef = null
44       if (product.image) {
45         imageRef = await uploadImageToSanity(product.image)
46       }
47       const sanityProduct = {
48         _type: 'product',
49         name: product.title,
50         description: product.description,
51         price: product.price,
52         discountPercentage: 0,
53         priceWithoutDiscount: product.price,
54         rating: product.rating?.rate || 0,
55         ratingCount: product.rating?.count || 0,
56         tags: product.category ? [product.category] : [],
57         sizes: [],
58         image: imageRef ? {
59           _type: 'image',
60           asset: {
61             _type: 'reference',
62             _ref: imageRef,
63           },
64         } : undefined,
65       }
66       console.log('Uploading product to Sanity:', sanityProduct.name)
67       const result = await client.create(sanityProduct)
68       console.log('Product uploaded successfully: ${result.id}')
69     }
70   } catch (error) {
71     console.error('Data import completed successfully!')
```

Now, let's install the necessary packages. Run the following command in your terminal:

```
npm install @sanity/client
```



5. Running the Import Script

To run the import script, we need to add a new script to our

`package.json` file. Open your `package.json` and add the following to the `"scripts"` section:

```
"scripts": {  
  
  "dev": "next dev --turbopack",  
  
  "build": "next build",  
  
  "start": "next start",  
  
  "lint": "next lint",  
  
  "import-data": "node scripts/importSanityData.mjs"  
  
}
```

Now you can run the import script using:

```
npm run import-data
```

This script will fetch products from the FakeStoreAPI, upload any associated images to Sanity's asset store, and then create new product documents in your dataset sanity.

